Hands On Code Practice
with ITensor

Table of Contents (click to jump to each hands-on):

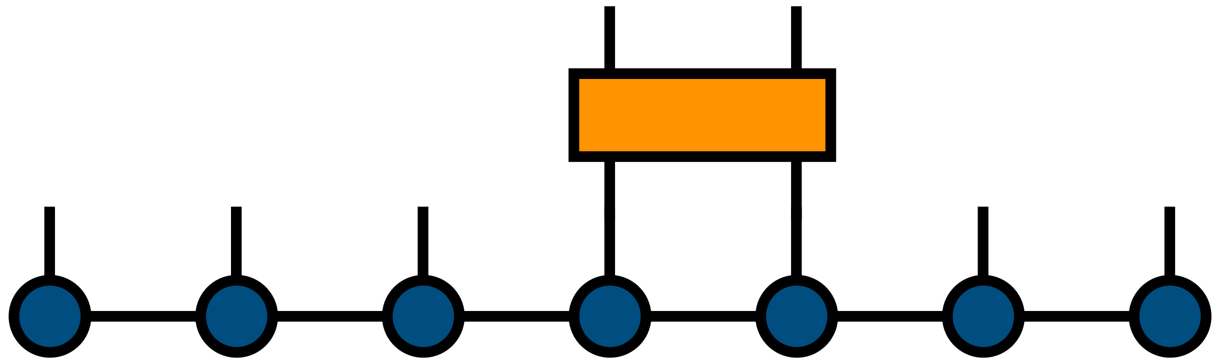
The TEBD (Gate Evolution) Algorithm – Hands On

Located in codes/gate_evolution/ folder.

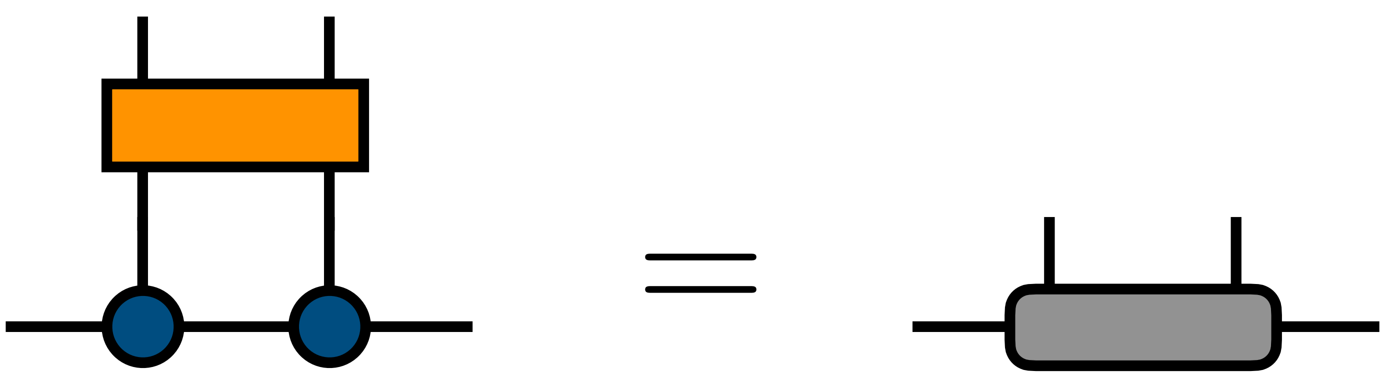
In this exercise, we will finish implementing a code for applying quantum gates to MPS using "TEBD"

The TEBD (Gate Evolution) Algorithm

Acting a <u>two-qubit gate</u> on a wavefunction in MPS form

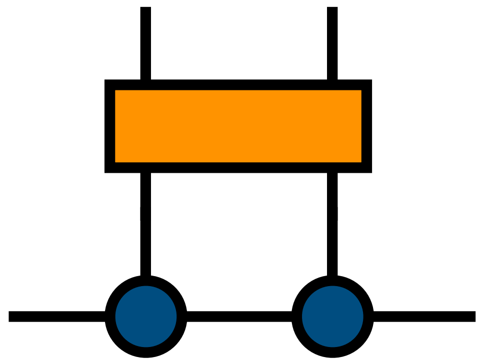


Operate on two tensors:



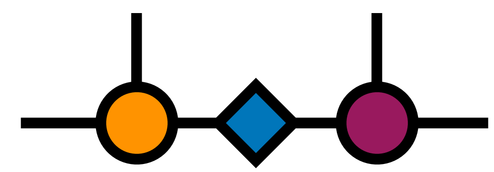
destroy MPS form locally

Recover MPS form using truncated SVD:



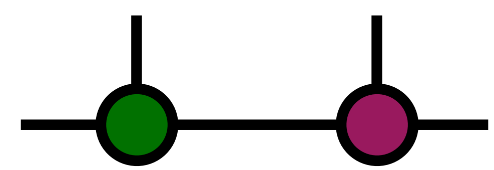
$=$

$\overset{\text{SVD}}{\simeq}$

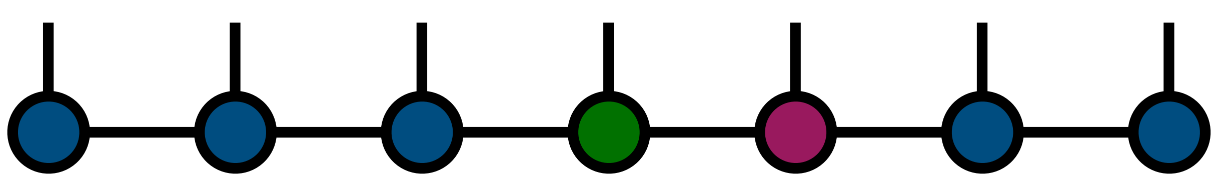


keep top χ *values*

$=$



Result:

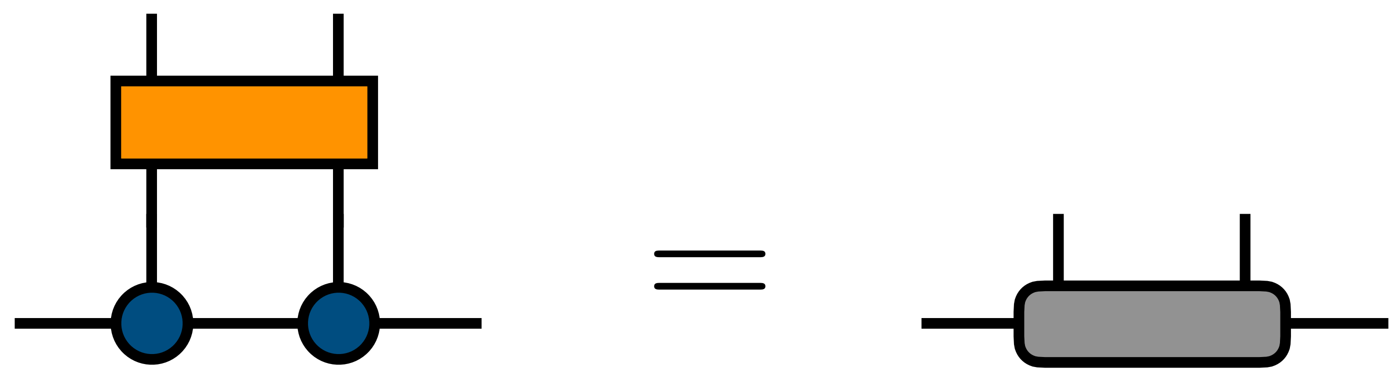


same bond dimension, small loss of fidelity

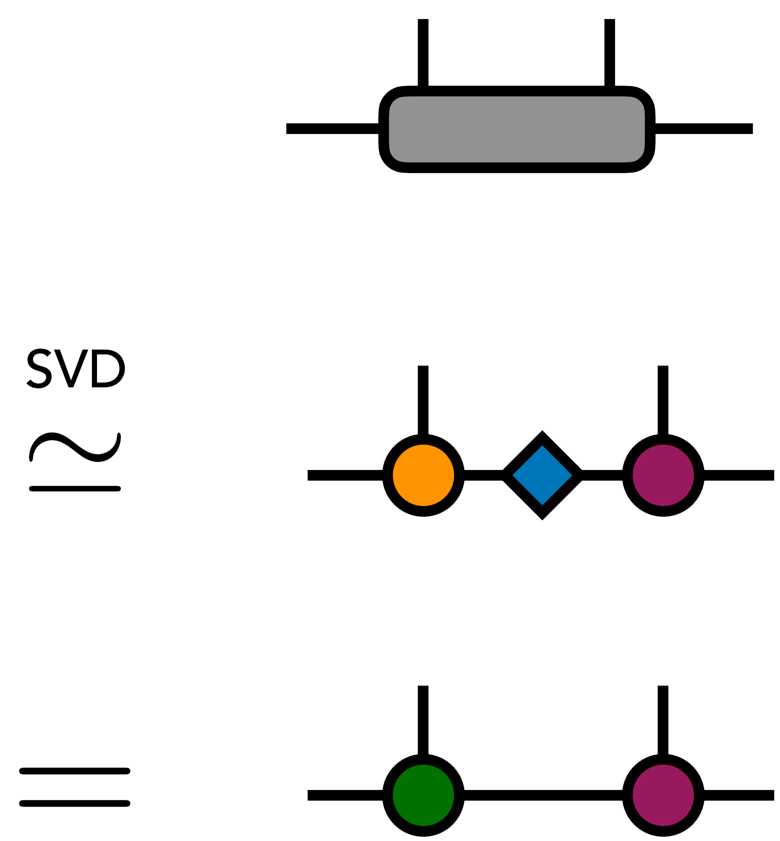
The TEBD (Gate Evolution) Algorithm – Hands On

Your task: implement the core steps of TEBD

(1)



(2)



The TEBD (Gate Evolution) Algorithm – Hands On

To check your work

- run the check_fidelity.jl code which will compare the state made by your code to a (more expensive) full-state simulation

- print the bond dimension (size of new MPS index) to make sure it is not growing exponentially, though it will still grow

The TEBD (Gate Evolution) Algorithm – Hands On

Further challenges:

- replace the randomly generated gates with gates of a specific quantum circuit
- put "swap gates" in between applications of local gates to implement long-range gates

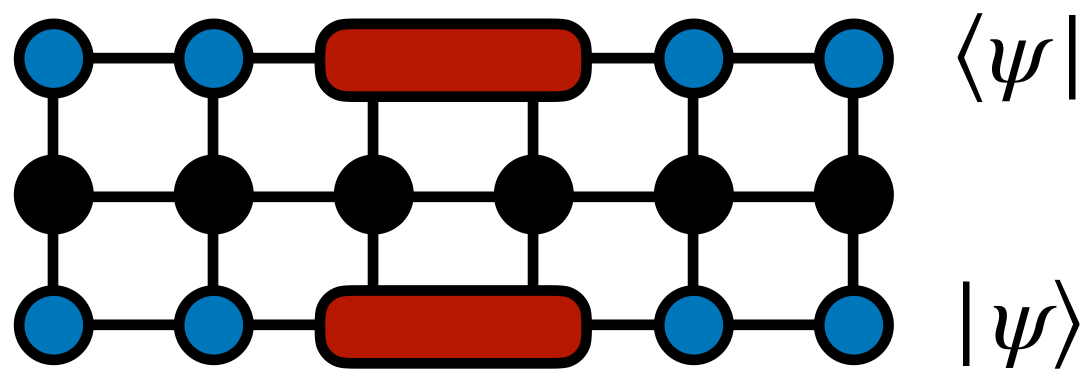
The DMRG Algorithm – Hands On

Located in codes/dmrg/ folder.

In this exercise, we will finish implementing a code for the DMRG algorithm

Specifically the *two-site* variant of DMRG

$E =$

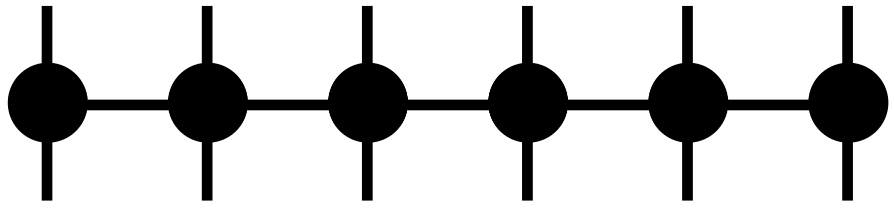


The DMRG Algorithm

DMRG algorithm

Assume we can write H as a tensor network

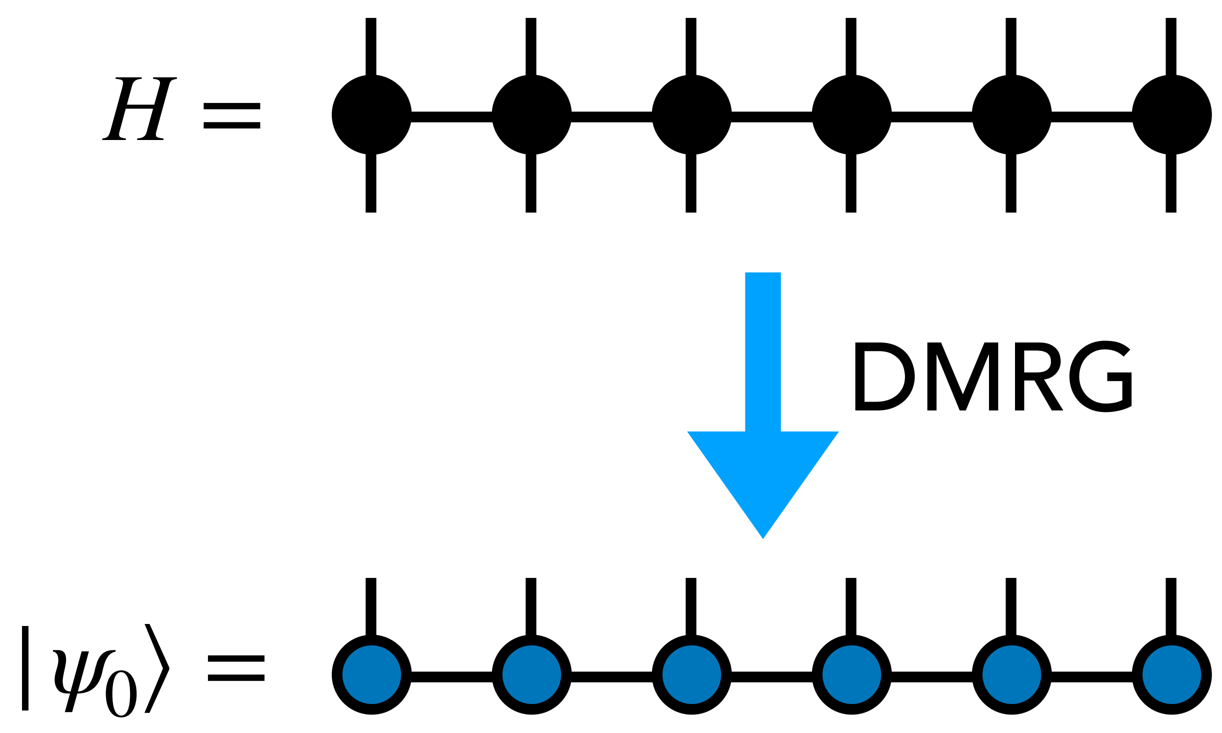
$H =$



The DMRG Algorithm

DMRG algorithm

DMRG finds its ground state (minimum-energy eigenvector) as an MPS tensor network

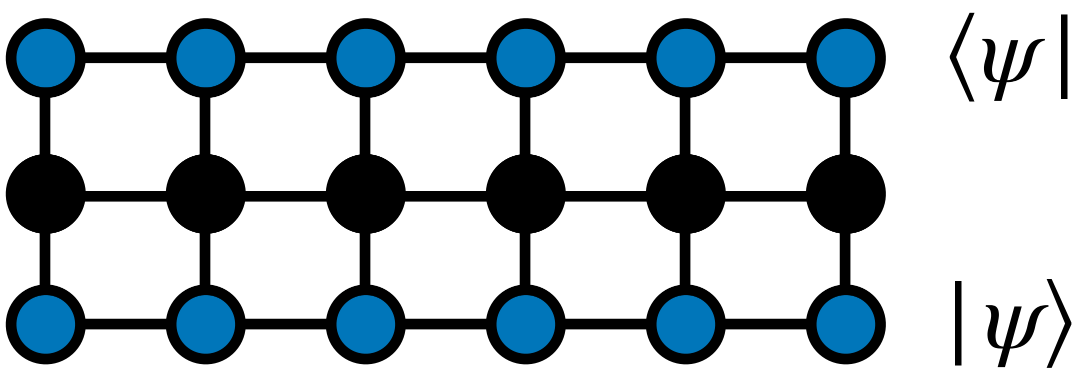


The DMRG Algorithm

DMRG algorithm

Energy is

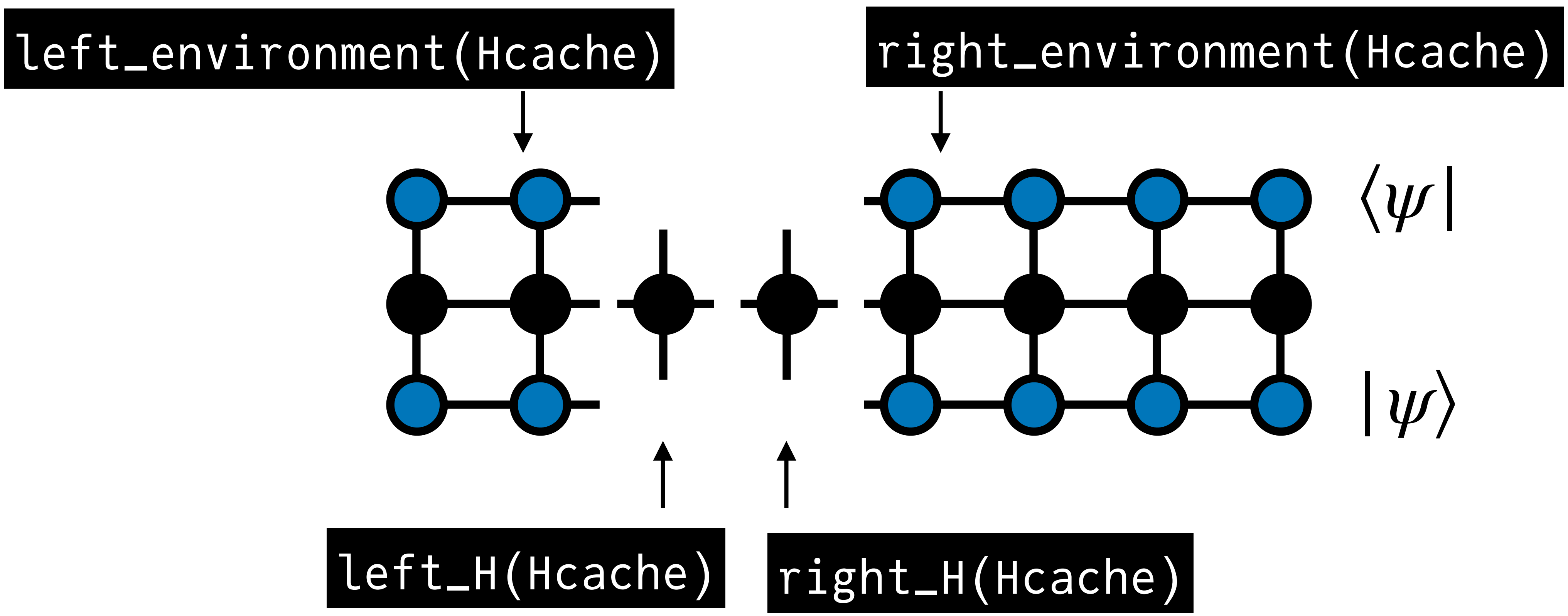
$E =$



The DMRG Algorithm – Hands On

Code includes a pre-written type called <u>MPOCache</u>

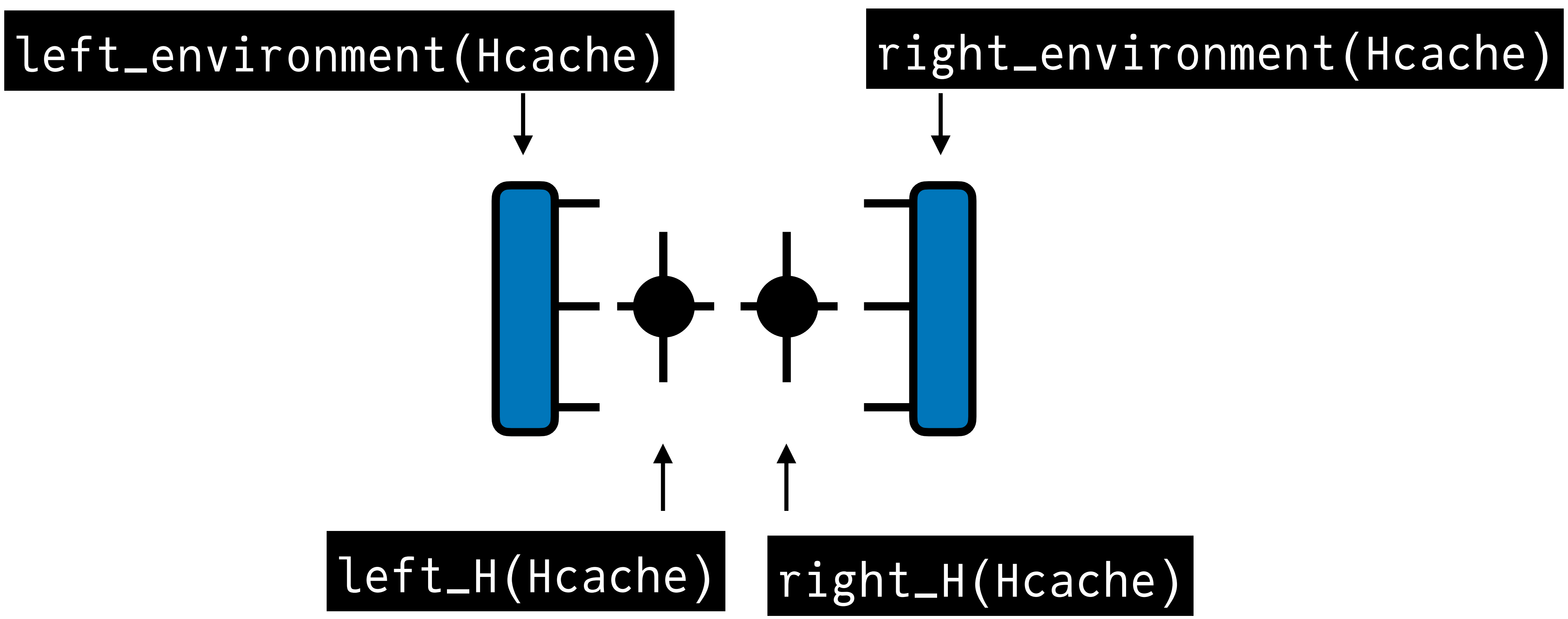
Calling `Hcache = position(Hcache,H,psi,bond)`
will build left and right environments around a two-site bond=(j,j+1):



The DMRG Algorithm – Hands On

Code includes a pre-written type called MPOCache

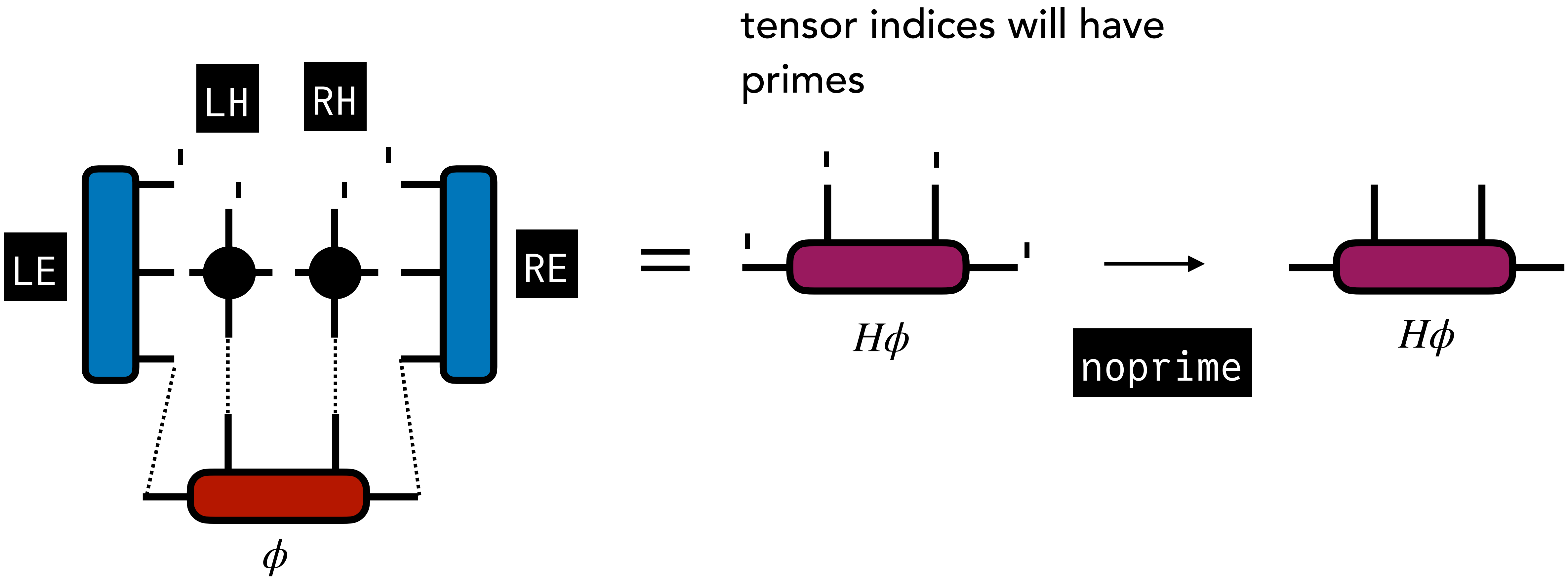
Calling `Hcache = position(Hcache,H,psi,bond)`
will build left and right environments around a two-site bond=(j,j+1):



The DMRG Algorithm – Hands On

Task 1: finish implementing the `mult(Hcache,phi)` function

What it should do:

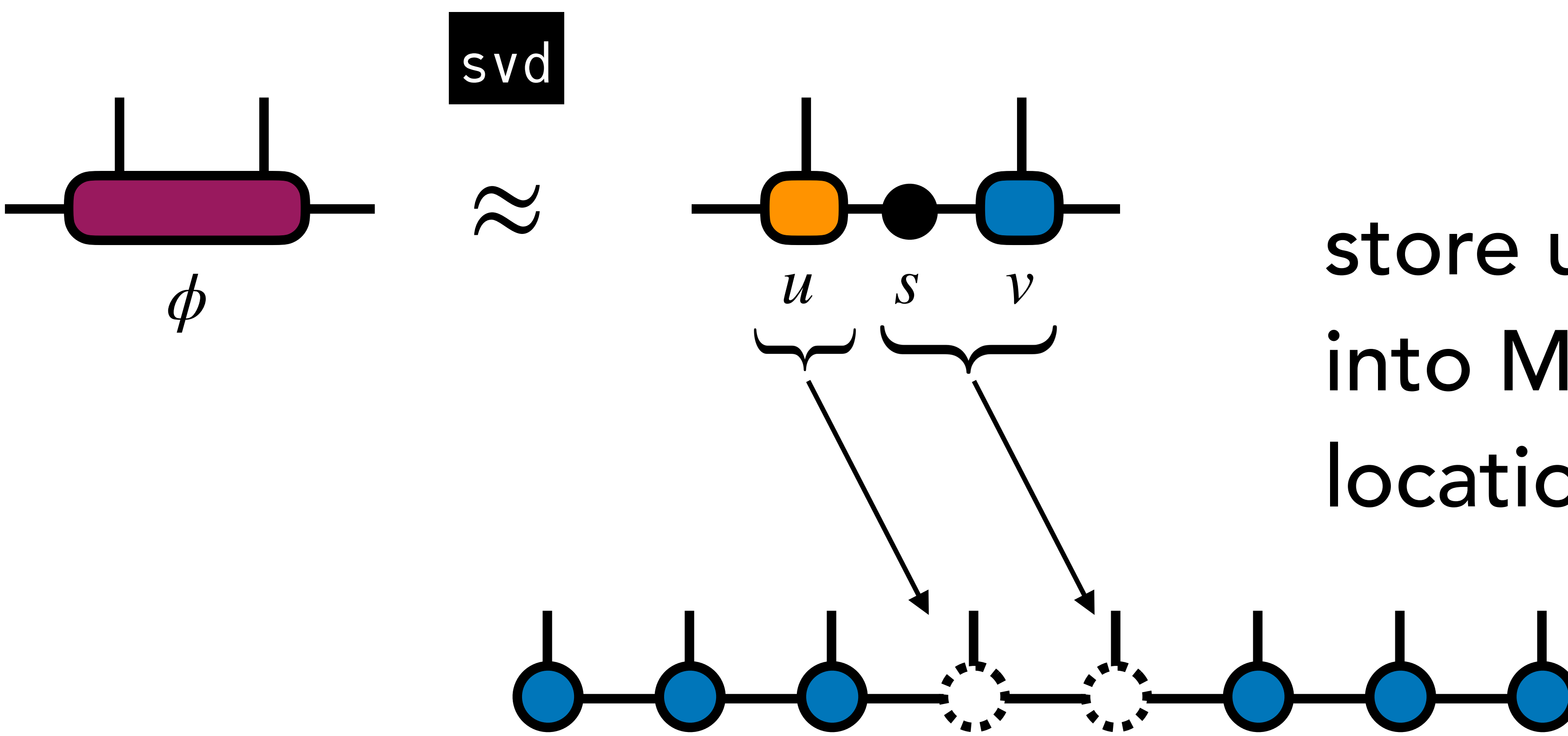


and return $H\phi$

The DMRG Algorithm – Hands On

Task 2: SVD apart the two-site wavefunction to put back into the MPS

What you should do:



store u and s*v into MPS in correct locations

The DMRG Algorithm – Hands On

Task 3: further improvements to your DMRG code

- check that energy is correct
- use the expect function compute expect(psi,"Sz") and plot. What is going on at the boundaries?
- improve printing options
  - print the bond the code is on
  - print the MPS bond dimension after the SVD
  - allow changing the amount of printing through a keyword option
- collect data during DMRG run (e.g. local properties of MPS) and use Makie.jl package to make a movie!

Functions as Tensor Networks – Hands On

Located in codes/function_integration/ folder.

In this exercise, you will use the TCI algorithm (from a package) to learn a function into MPS form, then write code to compute the integral $\int_0^1 f(x)\, dx$

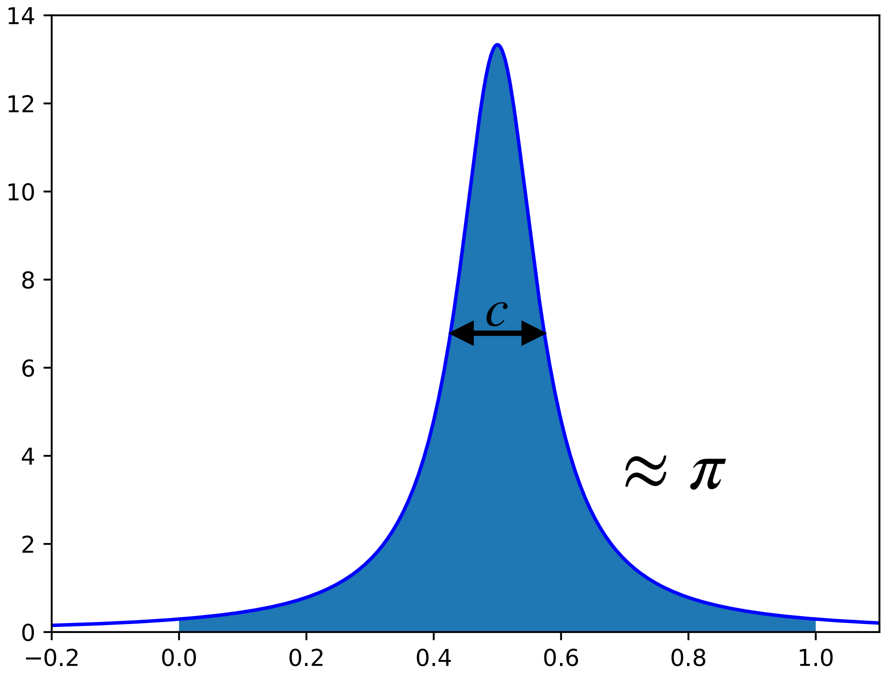
Applications of TCI

Target function: unnormalized Cauchy distribution

area under curve is π

$$f(x) = \frac{c}{(x - \frac{1}{2})^2 + c^2}$$

$$\lim_{c \to 0} \int_0^1 dx \; f(x) = \pi$$

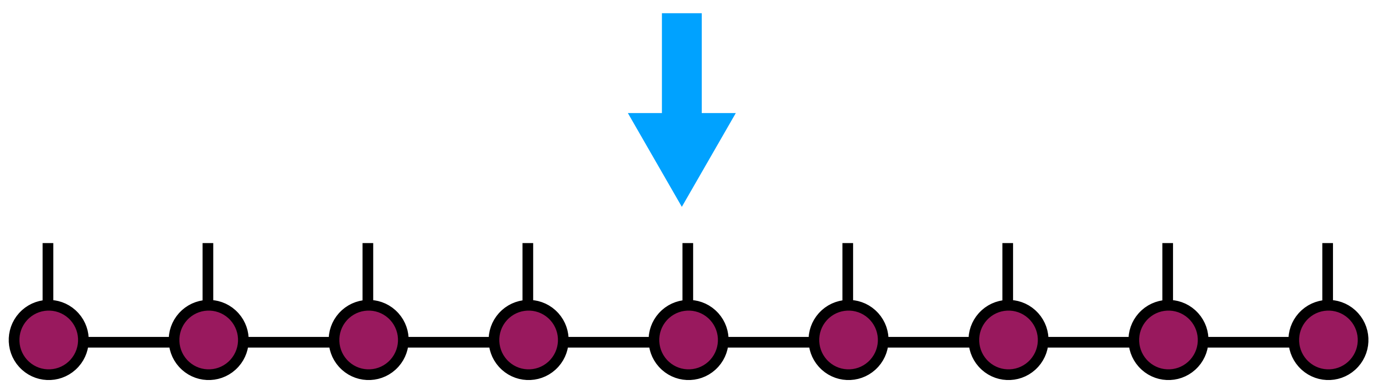


Applications of TCI

Once we have an MPS or "quantics tensor train" (QTT) version of the function

$$f(x) = \frac{c}{(x - \frac{1}{2})^2 + c^2}$$

$f(x) \approx$

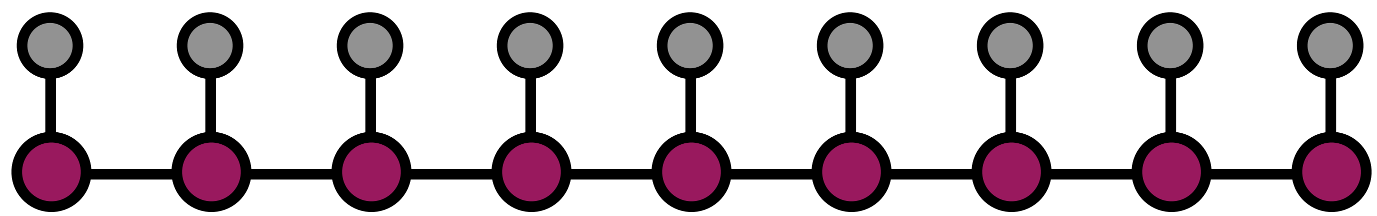


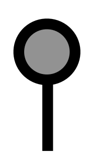
Applications of TCI

We can integrate it from [0,1)

by attaching "summation vectors"

$$\int_0^1 f(x)\, dx \approx$$





$$= \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

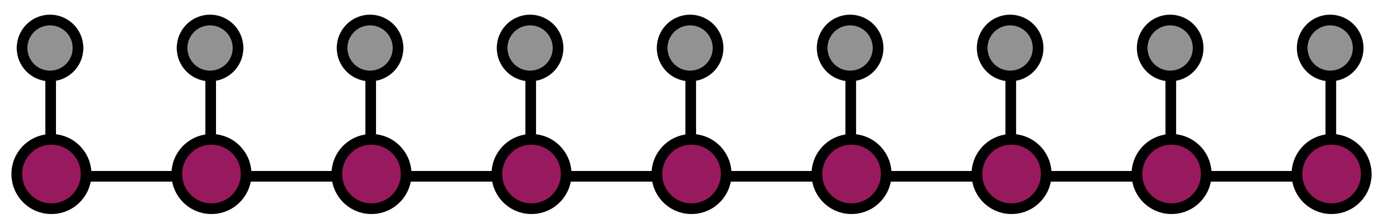
$1/2$ comes from the integral measure dx

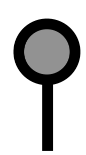
Applications of TCI

We can integrate it from [0,1)

by attaching "summation vectors"

$$\int_0^1 f(x)\,dx \approx$$





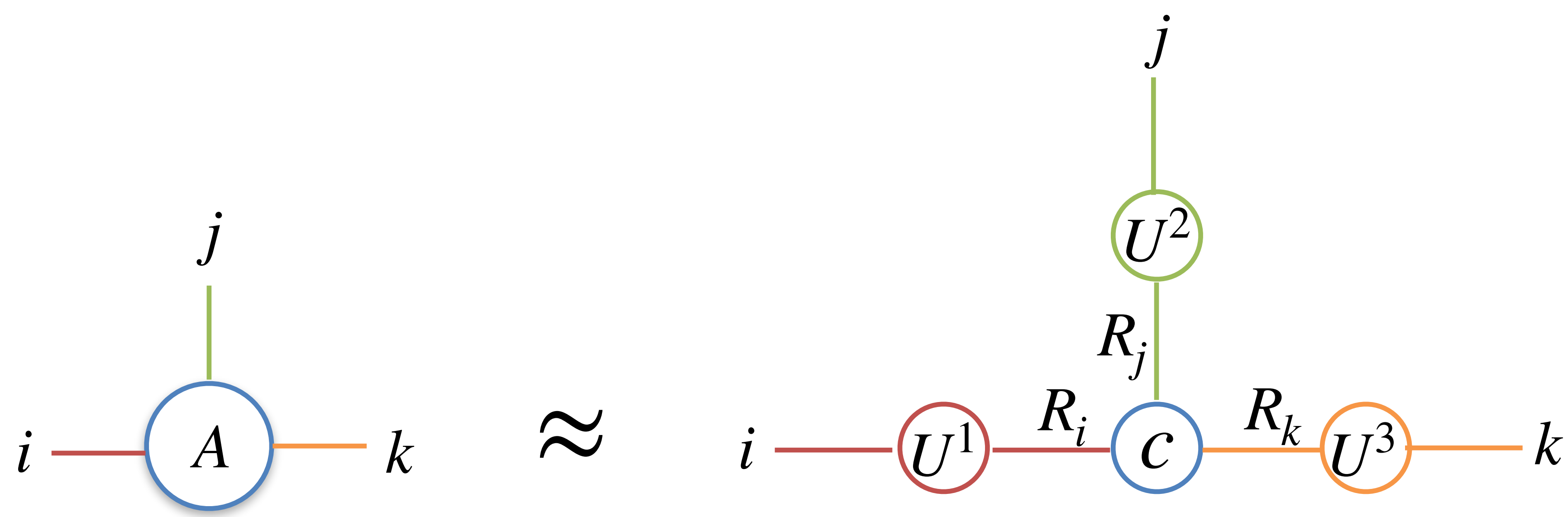
$$= \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

$1/2$ comes from the integral measure dx

<u>Task</u>: implement the integrate function to compute the above integral of an MPS function approximation

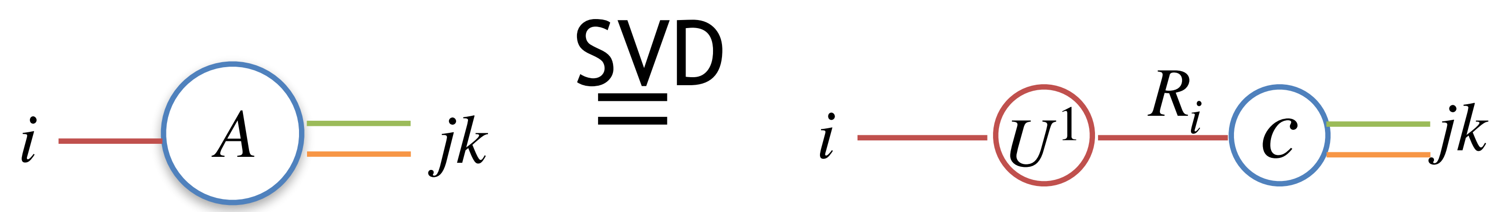
Live example: Tucker decomposition

Goal: represent a tensor A as a product of a compressed core tensor c and unitary transformation matrices U



Tucker decomposition

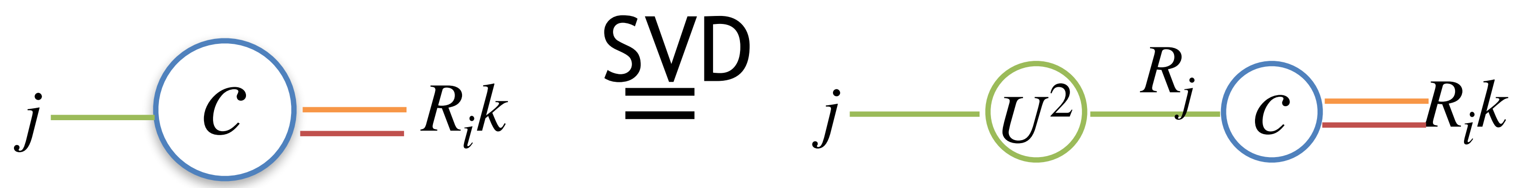
This can easily be done by systematically computing the SVD of each mode of the tensor A



This transformation becomes an approximation when the SVD is truncated based on the singular values of the decomposition.

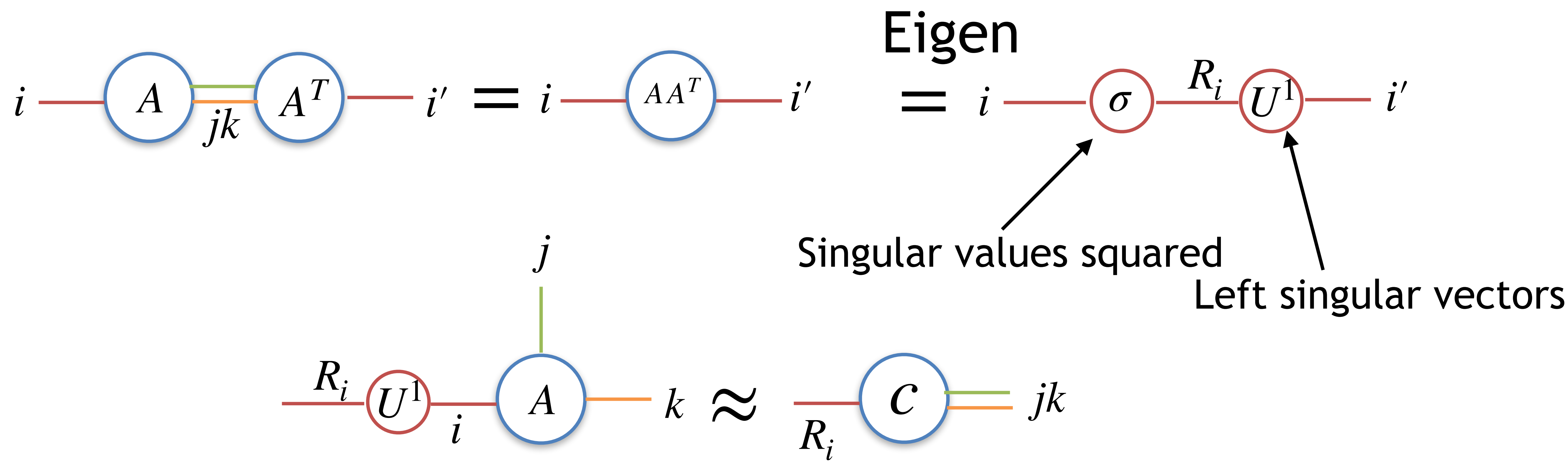
Tucker decomposition

Unfortunately, this method can be limited as it requires the SVD of relatively large and rectangular matrices.



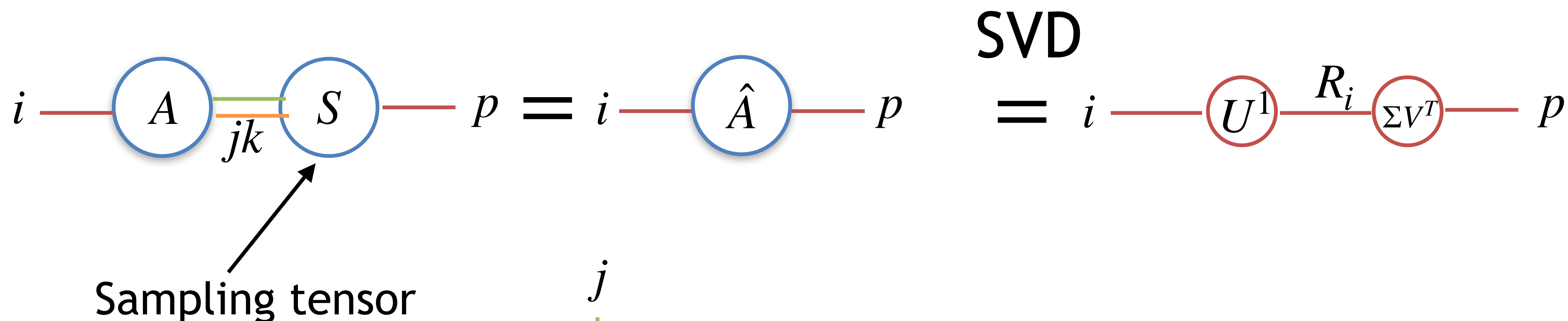
Fortunately, one can reduce the memory and computational intensity
by squaring original tensor

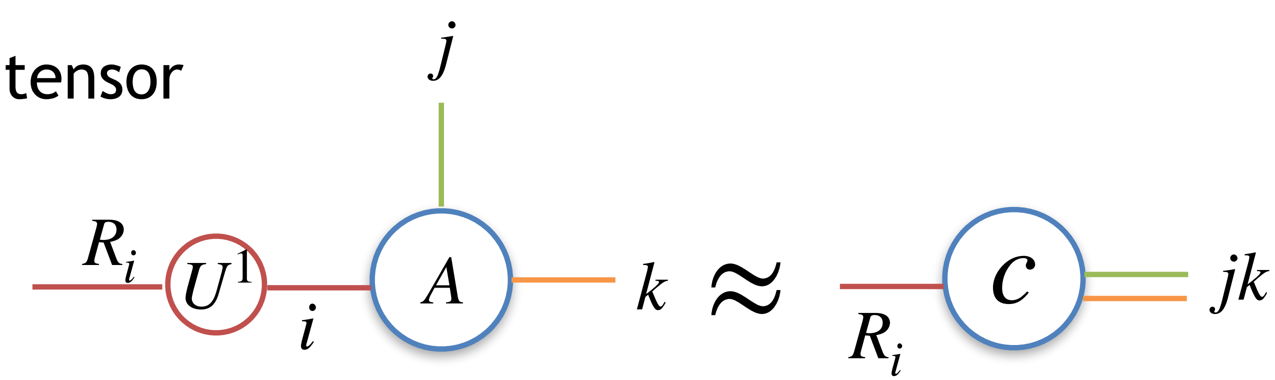
Smarter Tucker decomposition



Randomized Tucker decomposition

It should be possible to replace the squaring of A with randomized sampling.





HOOI optimization of tucker factors

There is no guarantee that the computed U matrices are the in fact the best set of Tucker transformations.

$$min_{C,\overrightarrow{U}}(\|A - C \times_{R_iR_jR_k} \overrightarrow{U}\|)$$

$$\overrightarrow{U} = [U^i, U^j, U^k] \qquad (U^p)^T U^p = I \in \mathbb{C}^{R_p \times R_p}$$

$$AU^jU^k \overset{SVD}{\approx} U^i_{R_i}\Sigma_{R_i}V^T_{R_i} = U^i_{R_i}C_{R_i}$$

https://arxiv.org/pdf/2110.12564